ARCOS – AI Rule-Constrained Orchestration System

1. Introduction

ARCOS (AI Rule-Constrained Orchestration System) is a schema-driven framework for orchestrating AI agents under strict specification and validation rules. It ensures that outputs are consistent, traceable, and domain-specific, using domain agnostic messaging, thus enabling reliable automation across software, hardware, and business domains.

By defining agent interactions through XML schemas (XSD), ARCOS provides a structured way to: capture user intent, translate it into rules, validate generated outputs, and ensure compliance with constraints. The system creates a new field of schema-driven AI orchestration, making AI results trustworthy and auditable.

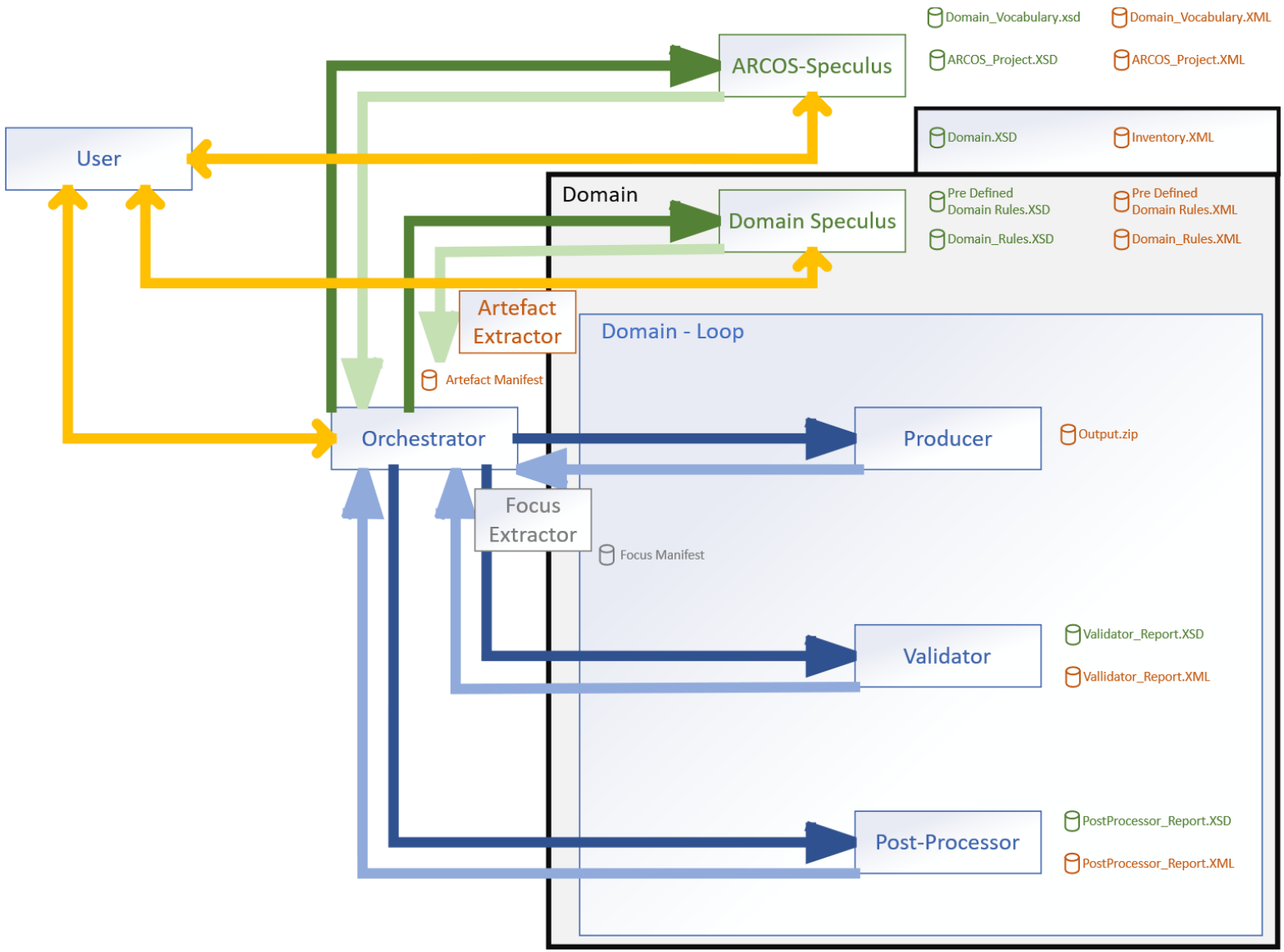
2. Architecture Overview

At its core, ARCOS operates as a loop between the user, the orchestrator (Maestro), and specialized agents: Domain-Speculus, Domain-Producer, Domain-Validator, and Domain-Post-Processor. Each agent communicates using XML defined by stable domain agnostic schemas.

The architecture ensures every message exchanged between components is validated against its schema, guaranteeing clarity, consistency, and the ability to detect errors early.

Key Roles:

- User – the Composer, providing high-level intent.
- Orchestrator (Maestro) – coordinates the interaction of all agents.
- ARCOS-Speculus – defines the Domain specific agents to use. It also gets a copy of the Domain Vocabulary, predefined rules to send to all Domain Agents.
- Domain-Speculus – extracts rules and constraints from user input generating the Current Project Rules.
- Domain-Producer – generates concrete outputs (e.g., code, configs, documents).
- Domain-Validator – checks outputs against rules, schemas, and constraints.
- Domain-Post-Processor – refines, compiles, or packages outputs.
- Filter-Inputs – reduce output.zip according to Validator/Post-Processor reports


Domain_Vocabulary.xsd
Domain_Vocabulary.XML
ARCOS-Speculus
ARCOS_Project.XSD
ARCOS_Project.XML
User
Domain.XSD
Inventory.XML
Domain
Domain Speculus
Pre Defined Domain Rules.XSD
Pre Defined Domain Rules.XML
Domain_Rules.XSD
Domain_Rules.XML
Artefact Extractor
Domain - Loop
Artefact Manifest
Orchestrator
Producer
Output.zip
Focus Extractor
Focus Manifest
Validator
Validator_Report.XSD
Vallidator_Report.XML
Post-Processor
PostProcessor_Report.XSD
PostProcessor_Report.XML

3. Messaging Flow

ARCOS agents communicate exclusively via XML documents defined by XSD schemas. This makes interactions explicit, testable, and auditable.

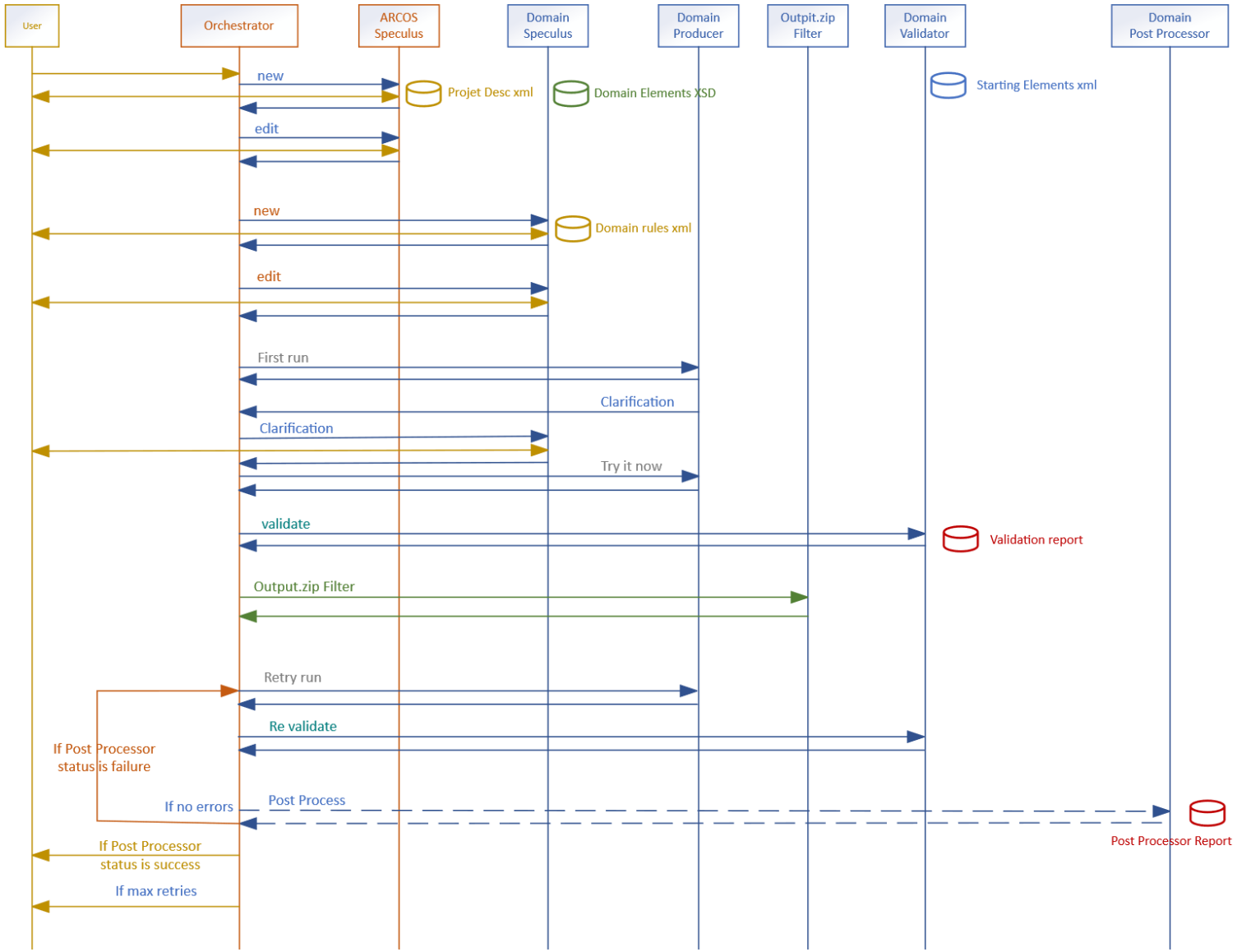
Typical flow:
1. The user starts a project (ARCOSProject.xml), using ARCOS-Speculus front end to define the associated Domain specific agents to use in the project and copies the Domain Vocabulary, predefined rules and Domain Schemas to send to all Domain Agents.
2. The Orchestrator sends the description to the Domain Speculus defined by ARCOS-Speculus. Maestro uses ARCOSDomainAgents.xml internally and forwards ARCOSProject.xml.
3. The Domain-Speculus extracts domain rules (Domain_Rules.xml).
4. The Domain-Producer generates initial outputs (ProducerResponse.xml).
5. The Domain-Validator checks outputs, returning ValidatorReport.xml or clarification requests,.
6. If successful, the Post-Processor finalizes outputs, generating PostProcessorReport.xml, and the output.zip is sent to the user.
7. If failures occur, the Orchestrator retries the series by sending to the Domain-Producer the previous output.zip with the report it received and clarifications can be requested until resolution.
8- Before sending to the Domain-Producer for a retry, the Orchestrator sends the output.zip and the report to a Domain-Filter so that the output.zip now only contains files named in the error reports. Reducing noise for the Producer to concentrate on what is wrong. The Domain-Filter builds the new zip and keeps what was good. At the end, when success is given by all agents, the Domain-Filter builds the complete output.zip for the user to receive, sending it first to the validator and postprocessor as complete output.zip to verify one last time that the full solution is valid.

4. Domain-Speculus and ARCOS-Speculus

The ARCOS-Speculus defines the domain specific agents to use for each component It also copies the Domain Vocabulary, the predefined domain rules, the Domain Schema to be sent by the orchestrator to all agents. It sends back to the Orchestrator the list of agents to call in its process and the DomainVocabularyAndRules. This makes ARCOS scalable across programming languages, industries, and domains.

The Domain-Speculus acts as the rules extractor for a specific domain. It translates raw user text into structured rules, validated against a schema. Each UserIntent is split into atomic rules, traceable back to its origin.

Domain-Speculus Workflow:
1. Human provides free-form description.
2. AI extracts UserIntents.
3. Rules are proposed, compared, and flagged as New/Update/Conflict.
4. Human reviews proposals.
5. Accepted rules are integrated into the canonical domain specification.
6. All rules are linked to their original UserIntent for full traceability.


User
Orchestrator
ARCOS Speculus
Domain Speculus
Domain Producer
Outpit.zip Filter
Domain Validator
Domain Post Processor
new
Projet Desc xml
Domain Elements XSD
Starting Elements xml
edit
new
Domain rules xml
edit
First run
Clarification
Clarification
Try it now
validate
Validation report
Output.zip Filter
Retry run
Re validate
If Post Processor status is failure
If no errors
Post Process
Post Processor Report
If Post Processor status is success
If max retries

5. Example Domain: BLEU Inventory

ARCOS has been thought with BLEU in mind, a Quebec replacement for Amazon, meaning a marketplace domain requiring precise inventory management.

Using the BLEU parts schema (BLEU_parts_v5.xsd), manufacturers define products such as bolts, nuts, washers, and packaged kits. Manufacturer in need to send to BLEU their offering, needs to build a software library to implement CRUD on any of its manufactured products defined in their Domain specific schema, using domain rules and predefined domain rules like CRUD and the language used (Rust).

Domain Vocabulary (Domain_Vocabulary.xml), Current Project rules (Project_Rules.xml) and predefined CRUD rules (Predefined_Domain_Rules.xml) ensure correctness (e.g., IDs unique, references valid, weights required, what you add you can delete …).

A Producer then generates a Rust CRUD interface for this domain, which the Validator checks against the rules. The Post-Processor compiles the Rust code and produces a report. This cycle illustrates ARCOS' ability to bridge high-level human intent to working, validated, domain-specific systems.

6. Benefits of ARCOS

ARCOS brings key advantages:

- Trustworthy AI Inter-communication: every message validated against schemas.
- Domain-agnostic: works with any domain that uses AI for its output.
- Traceability: rules linked back to original user intent.
- Extensibility: new domains can be added by publishing their specialized agents, be it a Domain-Speculus, a Domain-Producer, a Domain-Validator or a Domain-Post-Processor. Also publishing their Domain Vocabulary, Predefined Domain Rules and Domain Schema to be used by the Domain Producer.
- Reusability: the same orchestration loop applies across software, hardware, and business contexts.
- Filtering input files for more focus producer.

7. Conclusion

ARCOS formalizes how AI systems can be constrained, validated, and orchestrated through normalized messaging using schemas. By separating user intent capture, rule extraction, output generation, validation, and post-processing, ARCOS establishes a repeatable, trustworthy pipeline. This approach opens an entirely new field: schema: AI orchestration using schema-based messaging between agents and the Orchestrator.

ARCOS-Speculus

returns

ARCOSDomainAgents and

ARCOSProject

Orchestrator

Uses ARCOSDomainAgents

Sends ARCOSProject with its Vocabulary, Predefined Domain Rules and Domain Schema to the Domain-Speculus

Adds Current Project Rules if edit project rules

Domain-Speculus

Returns Current Project Rules

Orchestrator

Sends Vocabulary, Predefined Domain Rules, Domain Schema, and Current Project Rules to Producer

Producer

Returns Output.zip

Orchestrator

Sends Vocabulary, Predefined Domain Rules, Domain Schema, and Current Project Rules, and Output.zip to the Validator

Validator

Returns Validation Report

Orchestrator on validation success

Sends the Output.zip to the Post Processor

The current Post processor does not need Vocabulary, Predefined Rules, Domain Schema, and Current Project Rules

Post-Processor

Returns

Post-Processor report